Rapport de Projet
OCR Choucroute
par Nitojat

Guillaume Roffe
Michaël Willame
Cindy Lim

12 Décembre 2012

Table des matières

Chapitre 1

Introduction

Il faut beaucoup de courage pour admettre sa défaite, admettre qu'en fin de compte on a pas pu réaliser ce qu'on avait prévu.
Il y a un moment où on sait, on sait que même si on essaie, de toute façon on arrivera pas jusqu'au resultat escompté.
Lorsqu'on arrive au moment fatidique ou l'on ne peut plus rien faire, soit on cache ses erreurs (ce que beaucoup font), soit on les assume.

Effectivement Nous avont échoué, pris par le temps, les moyens. Nous n'avons pas atteint tous nos objectifs, mais nous avons quand même progressé.
Un certain nombre de question exigent encore une réponse :
Jusqu'ou en est on allé ? somme-nous vraiment si loin du resultat final ? Pourquoi avons-nous échoué ? comment faire pour que cela ne se reproduise pas ? C'est ce que nous vous dirons dans ce rapport.

Si chacun a un avis particulier sur la question, un consensus se forme autour du fait de désigner comme cause le sous-effectif flagrant dans lequel nous étions.

Chapitre 2

Organisation

2.1 Reprise du cahier des charges

Pour le projet et conformément aux exigences du cahier des charges nous devions fournir :

Pour la première souteance :

- Le chargement de l'image
- Les passages de couleurs en gris puis en noir et blanc
- L'élimination du bruit
- La rotation d'une image à partir d'un angle donné et la détection de l'angle nécessaire au redressement
- Une détection des zones de textes et une ébauche du découpage des lignes et des caractères
- Etudier le réseau de neurones
- Un site web

Pour la souteance finale :

- La finalisation du pré-traitement de l'image
- L'OCR doit savoir extraire les caractères et détecter les lignes de l'image
- La mise en place du réseau de neurone avec l'apprentissage et la sauvegarde
- Une interface graphique à laquelle y est intégré le traitement de l'image dans son ensemble, un correcteur orthographique et une sauvegarde des résultats
- Un manuel d'utilisation (et une aide contextuelle) et d'installation

2.2 Répartitions des tâches

2.2.1 Pour la première soutenance

	Guillaume	Michael	Cindy
Binarisation		x	
Rotation	x		
Segmentation		x	x
Recherche sur le réseau de neurones			x
Site web			x

Toutes les tâches ont été réalisées pour la premières soutenance.

2.2.2 Pour la soutenance finale

	Guillaume	Michael	Cindy
Amélioration de la segmentation		x	
Extraction des caractères		x	
Réseau de neurones	x		
Reconaissance des caractères	x		
Interface graphique			x
Correcteur orthographique			x
Sauvegarde	x		
Site web			x
Manuel d'installation et d'utilisation		x	

Malheureusement, contrairement à la dernière soutenance, tout n'a pas pu être fait dans les temps. A ce jour, il nous manque l'implémentation du réseau de neurones dans le traitement des images et donc l'apprenstissage et la sauvegarde qui va avec.

Chapitre 3

Le pré-traitement de l'image

3.1 La binarisation

Cette partie fut assez simple, les niveaux de gris ayant été faits en TP. Il s'agissait simplement de remplacer chaque pixel par un pixel ayant les 3 mêmes niveaux de couleurs (niveaux calculés grace a une formule fournie en TP).
Cependant, plutôt qu'une binarisation classique où l'on se contente de fixer un niveau de gris qui determine si un pixel est noir ou blanc, j'ai préféré une approche plus fiable. En effet, la technique employée est de prendre un échantillon du fond de l'image et de comparer la couleur de chaque pixel a cette couleur de fond selon la distance entre les deux (entendez la valeur absolue de leur différence) on affecte a un pixel les valeurs (255,255,255) (blanc) ou (0,0,0) (noir) a l'aide de la fonction get_pixel_color du module SDL. Il est a noter que les images étant déja en niveau de gris, il était inutile de tenir compte des trois valeurs de Rouges Vert et Bleu pour effectuer les test sur cette étape,ces trois valeurs étant identiques.

Pour cette deuxieme soutenance rien ou presque n'était a faire sur cette partie. Il a juste fallut une fonction de reduction de bruit supplémentaire pour gérer les artefacts de rotations et autres.
le principe est simple on parcours l'image et pour chaque pixel on vérifie un a un les pixels autour et s'il sont tous d'une autre couleur, on change le pixel considéré en conséquence. a noter que ce traitement se fait sur une matrice de booleens, en effet nul besoin d'une image pour cela.

3.2 La rotation

3.2.1 Méthode de Rotation

Dans le projet, ma tache a été d'effectuer une rotation d'une image avec un angle donné et détecter le dit-angle de rotation dans le cas où l'image ne serait pas droite. Pour cela j'ai dans un premier temps rechercher comment tourner une image en identifiant l'image comme une matrice de points et en calculant les nouvelles coordonnées de tous les points afin de les ranger dans une nouvelle matrice, qui devait représenter l'image ayant effectuée la rotation. J'ai découvert qu'il fallait mieux effectuer une rotation en prenant le centre de l'image comme origine, et de là calculer les nouvelles coordonnées. Les équations des calculs étant les suivants :
x' = x*cos(ang) - y*sin(ang)
y' = x*sin(ang) +y*cos(ang)

Le problème avec ces deux équations c'est que la rotation s'effectue par rapport à l'origine de l'image soit le point (0,0) au coin en haut à gauche de l'image, ce qui aurait pour effet de rogner les bords de l'image et donc de perdre énormément de texte. Pour palier à ce problème j'ai comme je l'ai dit précédemment appliqué une rotation en prenant le centre de l'image comme centre de la rotation. Les nouvelles formules sont alors celles-ci :
x' = centre + (x - centre) * cos(ang) - (y - centre) * sin(ang)
y' = centre + (x - centre) * sin(ang) + (y - centre) * cos(ang)

Afin d'éviter les calculs de débordements j'ai considéré la rotation par le centre comme la plus simple et la plus rapide à mettre en œuvre étant donné le temps imparti pour le faire, je voulais essentiellement me concentrer sur la détection de l'angle de rotation. Cependant d'autres méthodes existent pour éviter de rogner l'image comme effectuer un calcul permettant d'évaluer le débordement et de décaler le texte en conséquence, mais cette méthode était trop longue à mettre en action.
La rotation en elle même était donc faîte à ce point, avec ces équations j'ai pu calculer les nouvelles coordonnées des points, cependant il n'était pas utile de calculer toutes les nouvelles coordonnées pour tous les points de la matrice, en effet l'algorithme de binarisation et d'affaiblissement du bruit précède l'algorithme de rotation, on peut donc ne prendre en compte lors de la rotation que les pixels noirs et assigner leurs coordonnées dans une nouvelle matrice remplie de pixels blanc, ou d'entier égal à 255 dans le cas présent.

L'algorithme de rotation et de détection était alors clair dans ma tête, il fallait transformer l'image passée en paramètre en matrice rempli seulement de 255 et de 0, représentant respectivement le blanc et le noir. Effectuer la détection de l'angle puis faire une rotation de la matrice et enfin transformer à nouveau la matrice résultante en image à afficher à l'écran.

3.2.2 Méthode de détection de l'angle

Une fois l'algorithme de rotation écrit et testé, il restait la majeure partie du travail à réaliser, détecter l'angle par lequel il fallait tourner la matrice.

Après moult recherches et incompréhensions sur les différentes méthodes de détection, j'ai constaté que la méthode basée sur le Principe de Hough était celle qui revenait le plus souvent et qui semblait la plus efficace, cependant vu que je n'ai strictement rien compris à ce principe après plusieurs recherches sur internet, j'ai catalogué ce principe de détection comme étant valable pour une amélioration d'algorithme mais qui était horse le groupe a été créé de manière tardive étant donné qu'il se composait à la base de deux personnes et non quatre.

Pour revenir où j'en étais, j'ai décidé de détecter l'angle de rotation de propos dans cette première soutenance étant donné le peu de temps dont nous disposions pour réaliser ce projet, je rappelle qud'une manière différente mais qui m'étais compréhensible et rapide à effectuer. J'ai réfléchi pas mal de temps et décidé de la matrice.
Une fois que j'ai mon tableau représentant le nombre de pixels noirs par ligne, je calcule la moyenne de pixels n'effectuer un calcul de variance sur la matrice.

Dans un premier temps, je compte sur la totalité de la matrice le nombre de pixels noirs par ligne que je stocke dans un tableau de dimension la hauteur de doirs puis la variance du tableau, j'ai ainsi la répartition par ligne du nombre de points noirs, le principe est le suivant, la variance du tableau est maximale pour un nombre très important de pixels noirs sur certaines lignes et des lignes où il n'y aurait aucun pixels noirs.

Et c'est quand l'image est droite que c'est le cas, si l'image est de travers la répartition de pixels noirs par ligne est plus homogène et donc la variance diminue. Elle est maximale pour une image totalement droite. Je calcule donc la variance de ma matrice de départ puis j'effectue une rotation de la matrice vers la droite de 1.

Enfin je calcule à nouveau la variance de ma nouvelle matrice, et je continue de manière itérative jusqu'à ce que la nouvelle variance soit inférieure à la précédente, à ce moment je vérifie que j'ai bien tourné sur la droite, si la variance a effectivement augmenté, c'est qu'il fallait tourner sur la droite et j'ai alors trouvé la meilleure variance et donc le meilleur angle possible, dans le cas contraire il fallait alors tourner vers la gauche et je recommence mes calculs dans le sens contraire afin de trouver l'angle.
Après avoir tourné ma matrice dans tous les sens et trouvé l'angle. J'effectue une dernière rotation de ma matrice avec l'angle obtenu, la matrice résultante est alors transformée en image puis affichée. Ce sera cette image qui servira à la segmentation de l'image.

Une amélioration du code serait de directement récupérer la matrice une fois la meilleure variance trouvée ou encore de passer sur la méthode de détection de Hough, ce qui sera probablement effectué lors de la seconde soutenance. La méthode choisie est fonctionnelle bien qu'un peu longue.
Pour un gain de qualité de l'image, nous pensons effectuer la rotation sur l'image originale et non l'image binarisée (que nous avons utilisé pour découvrir l'angle) et ensuite effectuer la binarisation sur cette image tournée. Pour des raisons de deadline urgente, nous n'avons finalement pas eu le temps d'effectuer les changements de la rotation escomptés lors de la première soutenance.

Chapitre 4

Analyse de l'image

4.1 la segmentation et l'extraction de caractères

Deux techniques ont étées envisagées afin de reconnaitre les zones de texte.
La première était de constituer des liste de pixels noirs contigues afin de constituer plus tard des blocs. Afin de représenter ces blocs, le type *Charact* fut créé (il doit son nom au fait qu'il possede les même characteristique que les caracteres a etudier plus tard).
Ce type est composé d'une origine, d'une taille et de la liste des coordonées des pixels noirs du bloc par rapport a l'origine dudit bloc.
Cependant cette méthode avait un Handicap majeur : le facteur temps.
En effet si elle était d'une redoutable précision (les zones étaient détectées a la ligne près) elle nécéssitait le parcours simultané d'une image et d'une liste de liste(pour une complexité totale de l'ordre de n puissance trois ou n est le nombre de pixels dans l' image) et nécéssitait de constituer les blocs a posteriori (avec une complexité de l'ordre de n au carré), ce qui au total prennait, pensait-on plus de vingt minutes sur les plus grosses images de test.
Ainsi une autre technique fut mise au point.
La technique actuelle constitue directement les *Charact* en testant si les pixels s'inscrivent dans ou sont suffisamment proche d'un rectangle dans lequel s'inscrit un *Charact*.
Cette technique est ainsi seulement de l'ordre de n au carré (avec la même définition de n) et ne nécéssite pas de second passage pour constituer les blocs.

Son handicap et sa précision, variable selon les images car elle s'appuie sur un seuil qui est fonction de la taille des caracteres du texte. Cependant même dans les cas les plus précis, les blocs formés sont de l'ordre du paragraphe.
Pour ces deux methodes, il fallait cependant éliminer les boites contenant un seul pixel, vestiges du bruit de fond, ou artefacts de la rotation.
A aussi été fait dans cette partie une fonction qui permet de visualiser les blocs reconnus en les surlignant en bleu (0,0,255).

L'amélioration de la précision ne fut pas chose aisée, car on avait toujours ces problèmes de temps, et ce même avec la version plus optimale de la premiere methode de segmentation. Une étude plus pousée nous a permit de découvrir que la fonction qui prennait le plus de temps était la fonction d'affichage. Elle prenait autant de temps sur la version optimisée à cause d'une erreur bête, un problème de condition qui engendrait beaucoup trop d'artefacts. Des tentatives d'optimisation peu concluantes ont été faites et on est resté sur ce modele d'affichage, avec cependant l'ajout d'un fond de couleur duquel se detache mieux les pixels bleus qui surlignent les caractères. Dans une approche tres théorique, la fonction de segmentation utilisée se base sur la théorie des graphes :
en effet, on considère la matrice de booléens comme un graphe non-orienté. les points notés false (donc noirs) representent des sommets, et deux sommetss sont adjacents si et seulement ci les deux cases de matrice les représentant sont contigües. On extrait ainsi les composantes connexes de ce graphe. chacune représentant un caractere.
En pratique pour chaque pixel noir rencontré, on lance une procedure recursive qui si le pixel est noir lr stocke dans une liste le marque en blanc et réessaye sur se voisins.
Ainsi lorsque la procedure a terminé son travail, il ne reste plus qu'a stocker la composante dans un type charact et de passe au pixel suivant.
A noter qu'un même pixel ne peut se trouver dans deux charactères différents car chaque fois qu'on l'envisage après sa première rencontre, on l'ignore car il est devenu blanc, ce qui équivaudrait a marquer le sommet d'un graphe qu'on serait en train de parcourrir.

Ordonner les charactères : Une fois tous ces caractères récupérés, il faut encore les ordonner et les trier pour obtenir un texte cohérent.
Por cela on divise d'abord la liste de caractères qui forme notre image en sous-listes. Chacune de ces sous-listes représente une ligne et donc les caractères sont répartis en fonction de leur position verticale. Une fois ces listes obtenues, on les tries par position verticale croissante, puis on trie les caractères a l'intérieur par position horizontale croissante.
Une fois ceci fait, on divise les listes en mots en fonction de l'éspacement entre deux caractères consécutifs.
Une fois le traitement terminés, on obtiens donc 4 niveaux de listes imbriquées (texte, ligne , mot, caractère) prètes a envoyer au réseau de neurones. Ces fonctions n'on pas été entièrement et intégralement testées cependant du au manque de réseau de neurones.

4.2 Réseau de neurones

Généralités

Le fameux réseau de neurones, fut réalisé par moi même Guillaume.
Cette tâche, que je pense être la plus importante et la plus difficile dans le projet, vu qu'au final si on ne reconnait pas les caratères, le programme ne sert pas à grand chose, aurait dû selon moi être réalisé à deux et non pas un. Malheureusement, étant au nombre de trois et non de quatre dans le groupe, je me devais de le faire seul.
Après de nombreuses recherches sur le sujet, j'ai decidé d'utiliser un réseau perceptron à multi-couches. Je ne vais pas m'appesantir sur le fonctionnement du réseau perceptron, étant donné que je n'ai moi même pas réussi à comprendre totalement la finalité du fonctionnement du réseau.
Afin de produire un réseau fonctionnel j'ai choisi de réaliser un réseau "monocouche" soit une liste de neurones avec comme base qu'il y a un neurone pour chaque caractère, en se basant sur une probabilité de correspondre le plus possible à un schéma pré-enregistré. Ceci étant le modèle le plus simple à réaliser, vu que les autres modèles m'ont laissé mariner dans une forte incompréhension.

Fonctionnement théorique du réseau

Le réseau est censé fonctionner ainsi :
Après segmentation de l'image et des caractères en lignes puis mots. Je récupère toutes les boîtes créées, transforme le tout en matrice de booléens afin d'avoir une représentation de chaque lettre. Après redimensionnement des matrices, je les soumets à l'algorithme de reconnaissance de caratères.
je récupère ensuite les caractères résultats de la reconnaissance que je concatène afin de créer des mots, à cela que je les sépare ensuite par des espaces et ainsi de suite grâce à la distinction de chaque caractère faisant parti d'un mot ou d'un autre réalisée lors de la segmentation.
Ayant ma string résultat, la théorie voulait que je les donne ensuite à l'interface pour l'afficher puis la sauvegarder dans un fichier texte ou tout autre fichier de traitement de texte. Le problème étant que dans le temps imparti pour le faire, je n'ai pas réussi à terminer le réseau dans les temps. Et la puissance du langage utilisé était telle que je me suis cassé les dents à essayer de débuger ce réseau.
Pour une raison obscure la reconnaissance de caractères sur l'apprentissage ne fonctionna pas et de nombreux autres erreurs sont venues s'ajouter à cela. Je n'ai pas réussi à terminer le réseau cependant je pense que j'étais sur la bonne voix pour le faire, je n'en avais malencontreusement pas le temps.

Le réseau de neurones ne fonctionnant pas, la sauvegarde du dit réseau était rendu caduque. Ainsi je n'ai pas créé de fonctions pouvant réaliser la sauvegarde du réseau ou de l'image sélectionné.

4.3 Reconnaissance des caractères

Pour chaque neurone, une matrice de poids est associée, ainsi que le nombre de fois qu'il a appris un caractère spécifique. Lors de la reconnaissance on compare pour chaque pixel noir de l'image à reconnaitre si il possède un vis-à-vis dans la matrice de poids. Vu que pour certains caractères dans différentes polices la place d'une barre verticale peuvent ne pas varier, certaines parties sont molns importantes que d'autres, comme par exemple les polices avec ou sans serif.
Un poids est donc associé à chaque pixal afin de noter l'importance du point dans la reconnaissance du dit caractère, pour les besoins du code j'ai choisi arbitrairement de stocker les poids et donc de redimensionner les boites issues de la segmentation dans des matrices de 16 par 16.

Pour chaque pixel noir reconnu dans l'image on ajoute le poids du pixel correspondant à une somme qu'on divise à la fin par la somme des poids de la matrice de poids, on obtient alors une proabilité de'appartennce à tel ou tel classe decaratctère. On effectue cette opéraion sur chaque neurone et on ogarde le neurone et donc le caractère associé ayant le pourcentage le plus élevé d'être justement ce caractère. En prenant en compte un seuil à dépasser lors de la reconnaissance, on obtient un semblant de reconnaissance de caractère, le problème résidant dans l'apprentissage des neurones pour chaque caratère, c'est alors là qu'ont réellement commencé les problèmes.

L'apprentissage a suscité pas mal de discution, nous avons finalement choisi lors de l'apprentissage de donner au programme une image de teest comprenant tous les caractères à reconnaitre dans l'ordre afin de les segmenter et de les associer à chaque neurone pour créer la matrice de poids corresponante et "mouler " le caratère dans la matrice. Afin d'une reconnaissance plus fine il aurait fallu donner plusieurs images de la liste de caractères dans des tailles et polices différentes pour améliorer et étoffer le réseau, cependant vu que je n'ai finalement pas réussi à faire fonctionner la première version de cet algorithme, je n'ai pas poussé plus loin l'apprentissage de caractères.
J'ai utilisé un principe de rétropropagation simple lors de l'apprentissage, lors de l'apprentissage d'une nouvelle image, je timinuais ls poids correspondants aux points blanc de l'imgage mais possédant des poids strictement positifs. Utilisant un réseau simple la détection risquait de tendre vers un apprentissage par coeur sans rétropropagation possible à la fin de la reconnaissance, cela restait une des fonctionnalités à implémenter mais dont je n'ai pas eu le temps pour.

Chapitre 5

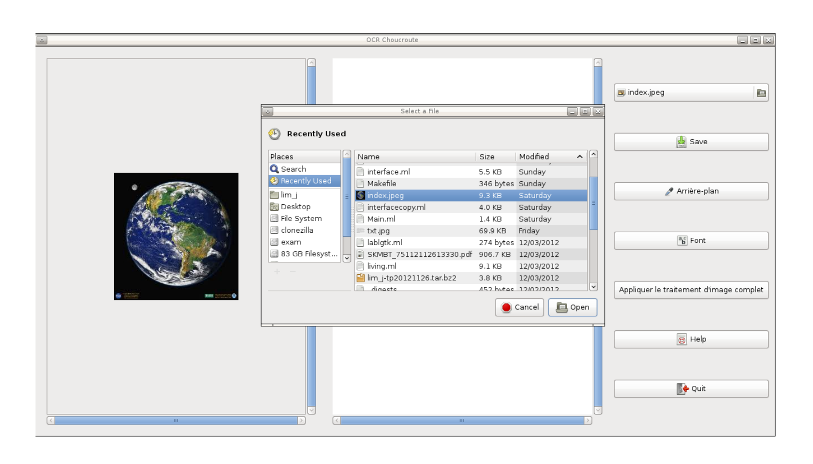
la partie interface et communication

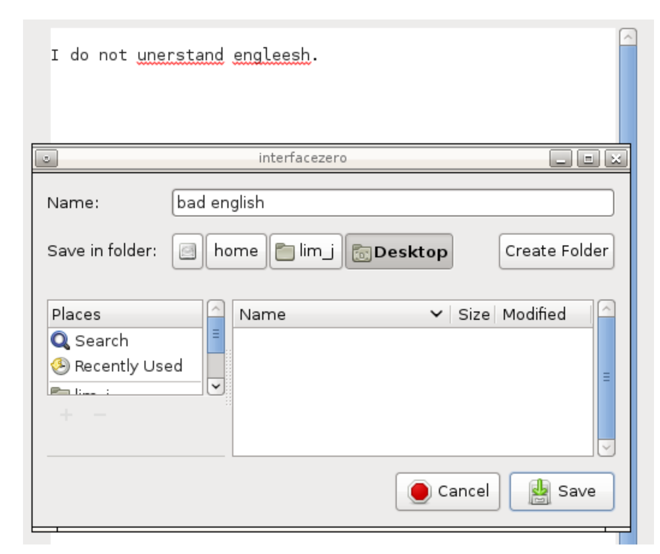
5.1 Interface graphique

Pour pouvoir intérargir plus aisément les fonctionnalités de notre OCR, nous avons créer une interface graphique en utilisant lablGTK dans ce but.

Comme exigé dans le cahier des charges, l'interface comporte deux fenêtres, une à gauche qui affiche l'image originale chargée depuis l'ordinateur qui restera tel quel même après traitement (binarisation, rotation, segmentation, extraction des caractères, reconaissance des caractères) .Celle de droite permet de modifier le texte du document numérisé (supposé) apparaitre après le traitement.
Mais ce n'est pas tout. L'interface possède d'autres fonctionnalités notamment celle de décider de la couleur de l'arrière-plan et modifier la taille et la police des caractères dans la fenêtre de droite. Il est aussi possible de sauvegarder le texte dans la même fenêtre. Un correcteur orthographique a été également ajouté depuis les bibliothèque de lablGTK pour proposé des solutions aux erreurs de syntaxes possiblements commises. Ce correcteur marche aussi dans la langue anglaise.

En résumé, notre interface graphique peut :
- Charger une image
- Effectuer un traitement complet de l'image
- Modifier le texte provenant de l'image après traitement ou en écrire un
- Sauvegarder le texte


OCR Choucroute
Select a File
Recently Used
Places
Search
Recently Used
lim_j
Desktop
File System
clonezilla
exam
83 GB Filesyst...
Name
Size
Modified
interface.ml 5.5 KB Sunday
Makefile 346 bytes Sunday
index.jpeg 9.3 KB Saturday
interfacecopy.ml 4.0 KB Saturday
Main.ml 1.4 KB Saturday
txt.jpg 69.9 KB Friday
lablgtk.ml 274 bytes 12/03/2012
SKMBT_75112112613330.pdf 906.7 KB 12/03/2012
living.ml 9.1 KB 12/03/2012
lim_j-tp20121126.tar.bz2 3.8 KB 12/03/2012
Cancel
Open
index.jpeg
Save
Arrière-plan
Font
Appliquer le traitement d'image complet
Help
Quit


I do not unerstand engleesh.
interfacezero
Name:
bad english
Save in folder:
home
lim_j
Desktop
Create Folder
Places
Search
Recently Used
Name
Size
Modified
Cancel
Save

5.2 Site web

Pour communiquer avec le monde, un site internet à sa plus grande importance. Voici l'addresse du nôtre :

http ://ocrchoucroute.wordpress.com/

Via ce site nous pouvons découvrir la Nitojah team et l'OCR Choucroute, les différents aspects du projet : de la binarisation au réseau de neurones, et d'autres choses comme vous pouvez le voir sur cette image :



Chapitre 6

Impressions

6.1 Guillaume Roffé

Ce projet a été pour moi une expérience extrémement enrichissante, dure, difficile, mais cela m'a permis de mieux comprendre comment réagir face à un problème très difficile à résoudre mais pas impossible, j'ai lamentablement échoué mais j'ai appris à travailler en équipe sous-pression, en très peu de temps. J'ai également appris à mes dépens le nombre d'heures de sommeils qu'il m'était nécessaire d'avoir afin de pouvoir travailler.

J'ai dans un premier temps réalisé lors du traitement de l'image la rotation et la détecion de la rotation de l'image que j'ai apprécié de faire. Puis pour la seconde soutenance j'ai été chargé de réaliser le réseau de neurones, cette tâche fut ardue, et m'a demandé beaucoup d'investissements.

Cela m'a pris beaucoup de temps de comprendre comment je pouvais réaliser cela. Après de nombreuses recherches j'ai compris qu'un quatrième membre n'aurait pas été de trop pour m'épauler dans ce travail, étant donné que Michaël et Cindy étaient occupés par leurs travaux respectifs j'ai dû me résoudre à le faire seul. Cela m'a permis de voir jusqu'où pouvait aller mes capacités de reflexion et de concentration.

N'ayant pas réussi à finir le réseau, j'étais déçu que mon travail ne serve finalement à rien. Nous aurions pu clairement faire mieux, mais faute de temps et de personnel il nous était impossible d'en faire plus sans un déficit mental conséquant et une pénurie de café.

6.2 Michael Willame

Au début du projet, j'étais éffrayé, oui éffrayé, par la quantité de travail demandée en si peu de temps. Mais, à force de travail acharné, il se trouve qu'on a réussi à obtenir pour la première soutenance quelque-chose qui tenait la route.
Le resultat, compte tenu du fait que nous n'étions que 3 était encourageant si bien que je sortis de la première soutenance confiant (beaucoup trop malheureusement).

Ce n'est que sur la deuxième partie que le manque d'un quatrième membre se fit vraiment sentir. J'ai choisi de ne pas prendre le réseau de neurones parce-que j'avais peur de me perdre dans les méandres du concept si j'éssayais de comprendre (ce qui est arrivé a Guillaume.) Alors qu'il aurait fallu mettre au moins deux membres sur le réseau de neurones, il y avait sur les autres parties suffisament de travail pour nous occuper, Cindy et moi même.
Ainsi du temps fut perdu et ne connaissant rien aux réseau de neurones faute de nous y être plongés dès le début nous n'avons pu aider Guillaume.
Le fait que cindy n'était pas dans notre classe était également handicapant, à cause des incompatibilités d'emploi du temps.
Cette situation me mit dans un etat de stress intense : celui de ne pas avancer et surtout de ne rien pouvoir faire. c'est donc dans un etat de stress que le rendu fut fait.
Pouvait-on faire mieux ? oui, mais pas sans sacrifier le reste de nos matières scolaires (avec ou sans chute de note d'assiduité) ou distordre l'espace temps d'une manière ou d'une autre.

A noter également que cette expèrience m'a permit de découvrir une chose : j'abhorre le traitement d'image (même si j'en apprécie les applications).
La partie segmentation ne m'a plue vraiment que lorsque j'envisageait ma donnée comme un graphe et non comme une image. Cependant, cette vision est uniquement basée sur une expèrience et est donc à remettre en question.

Si le projet n'est pas terminé, on a quand même un traitement d'image assez précis pour d'autres applications, compatibles elles aussi avec l'interface. Par exemple, on pourrait trouver et enregistrer toutes les images d'un document. sans beaucoup de modifications.

6.3 Cindy Lim

Après le projet de Sup, que j'ai bien aimé, s'attaqué au projet de l'OCR fut un véritable challenge. En effet, j'ai trouvé que le niveau était bien plus élevé du fait des contraintes (notamment les délais qui sont beaucoup plus court et de l'handicap du dernier membre manquant dans notre groupe) et des exigences du cahier des charges imposés. De plus, contrairement à l'année dernière, la documentation parfois nécessaire n'est pas aussi diverse et variée (ce qui fut le cas pour le réseau de neurones).Ceci dit, l'expérience fut plutôt intéressante parce qu'elle diverge beaucoup de celle vécu avec la réalisation du jeux vidéo.

Je suis consciente que notre produit final pour le projet est loin d'être le meilleur et ne correspond pas exactement au cahier des charges. Néanmoins nous avons fait de notre mieux pour répondre aux demandes et nous nous sommes tous appliqués de façon à peu près égale dans la réalisation de l'OCR. Nous ne pouvons peut-être pas prétrendre être fière du résultat, mais je pense que nous n'avons pas à avoir honte non plus car nous avons tous donnés.

Chapitre 7

Conclusion

Comme nous le disions dans l'introduction, il est dur d'admettre un échec mais il est cependant important de revenir dessus et de comprendre de quel manière nous aurions pu réaliser un programme fonctionnel avec ce que nous avions.

Un projet de détection de caractères a pour but de détecter des caractères, le nôtre détecte effectivement si notre image est droite ou non et la redresse le cas échéant détecte les caractères et les entoure afin de les distinguer, le problème étant qu'il ne fait pas plus et que le réseau de neurones ne fonctionne pas, le rendu final nous reste un peu en travers de la gorge. Nous aurions pu faire beaucoup mieux, avoir un réseau de neurones complet mais il nous aurait fallu plus de temps.

Au final, nous n'étions pas si loin de réussir, de nombreux éléments ont été implémentés mais ont nécéssité plus de temps que prévu pour les fusionner. Certains n'ont pas pû l'être à temps. Nous avons tenté, Nous avons échoué cependant nous avons beaucoup appris de ce projet et de cette échec. Afin de mieux nous relever et de nous préparer à de nouvelles épreuves, potentiellement plus dures.